Victorian 6502 User Group Newsletter

KAOS

For People Who Have Got Smart

OSI SYM KIM AIM APPLE UK101 ORANGE

Registered by Australia Post
Publication No. VBG4212

ADDRESS ALL CORRESPONDENCE TO 10 FORBES ST., ESSENDON, VIC. 3040

Vol.3 No.4 January 1983

Welcome to the first KAOS newsletter for 1983. We can't promise you an exciting year full of surprises as we never know from one month to the next what's going to happen. We just muddle along, somehow there always seems to be something new and interesting to write about, hopefully this will continue for this year.

Please remember that membership fees are now due. The mailing list will be updated on the 16th February and if we haven't received your fees by that date you will not receive the February newsletter. If you intend to stay in KAOS we would appreciate receiving your fees before the 16th as it is annoying to remove a name from the mailing list and then have to put it back a few days later, Marian Anear will join me in thanking you for your co-operation.

You may be aware that George (COMP-SOFT) has gone to Greece for a holiday. Some people were afraid that while he was there he would either get married or be conscripted into the Greek army. We have just received word that our fears were groundless, because of all the gym work he did before he left he was able to out-run the girls, and the army didn't want him: he failed the IQ test!

A member has suggested that we put area codes on all phone numbers, we are doing it in the newsletter and the heading on the front page will be changed as soon as possible, at the moment all these numbers have an 03 prefix. If you are writing material for the newsletter and want your phone number included, please put your area code on your number.

The next meeting will be at 2pm on Sunday 30th January 1983 at the Essendon Primary School, which is on the corner of Raleigh and Nicholson Streets, Essendon. As the children will still be on holidays the school will not be open untill 1pm.

Items for the February newsletter must be received by the 13th February

INDEX

SMART LISTER
by Kerry Lourash

Since OSI ROM BASIC allows only seventy-two characters in a BASIC line, it is often necessary to write code with no spaces between characters, which tends to produce illegible listings. SMART LISTER is a utility that operates as an improved LIST command, inserting spaces at strategic places in the BASIC lines it lists, in order to make the lines more legible.

I was impressed when I saw the APPLE's method of program storage. It removes non-significant spaces from BASIC lines when they are tokenized and adds extra spaces when the lines are listed. On closer inspection, however, the APPLE system is not completely satisfactory. It prints a space before and after each token in a line. There's nothing REALLY wrong with this method; it is simple and effective.

Like all programers, if I was going to create a system, I was going to do it my way. An APPLE listing is just a bit too spread out for my taste; I don't feel arithmetic operators (-, -, /, *) should be segregated by spaces. Also, in the APPLE system, when two keywords are adjacent, a double space separates them. APPLE doesn't check to see if the previous character was a space before printing another space. Since OSI doesn't screen spaces on input, I wanted to include a redundant-space check in my system.

HERE ARE THE RULES OF SMART LISTER:

1. Don't add redundant spaces.
2. Insert a space after every statement (colon).
3. Insert a space after every keyword with a token equal or less than the STEP token.
4. Insert a space before the TO, THEN, AND, STEP, and OR keywords.

SMART LISTER can be called as a USR routine. With X=USR(X) installed as line zero of a program, LISTER can be called with a RUN command. Higher line numbers will not be executed, since LISTER exits to the immediate mode after its work is done. The routine can be installed in any part of memory without modification of its code. LISTER occupies less than 300 bytes.

To use the routine, simply call LISTER and reply to the lower-case "list" prompt as you would type a list command.

Now what we need is an input filter to screen out spaces in a line before it is tokenized. Better yet, a space-gobbler program that deletes all unnecessary spaces from a program in memory. Don't count on me, though; look how long it took me to get around to developing SMART LISTER!

Next page please

FOR SALE

MISCELLANEOUS ASSEMBLED and TESTED TASKER BUSS BOARDS, including RAM cards $50, Floppy Disk card $35, Mother Board $50.

5V 10amp POWER SUPPLY, including all leads, nad 12V 5amp for 5 1/4" floppy disk drives. In case with keyswitch etc. $100.00

MODEL 15 TELETYPE, including manual, excellent condition. Has worked with Superboard. $70.00

All prices are negotiable. I also service and repair most OSI systems at very reasonable rates. Paul Dodd

```
; *SMART LISTER*
; by K.V.L.
;MOVEABLE CODE
;                   ROM       DISK
 BPTR       = $C3       $C7
 BUFFER     = #13       $1B
 CHRGET     = $00BC     $C0
 CHRGOT     = $00C2     $C6
 CHRSAV     = $13       $1B
 CTRLC      = $A629     $0819
 EVAL       = $A77F     $096C
 FINLIN     = $A432     $0633
 INPUT      = $A357     $0558
 LF:CR      = $A86C     $0A73
 NUMPRT     = $B95E     $1CDC
 OKAY       = $A274     $0474
 OUTPUT     = $A8E5     $0AEE
 POKER      = $11       $19
 QUOFLG     = $60       $10
 SPACE      = $20       $20
 TOKSAV     = $14       $1C
 TOKTBL     = $A084     $0284
 VARPTR     = $AA       $AC
 YSAVE      = $97       $96
;
;** TOKEN LABLES **
 AND.       = $48
 OR         = $A9
 STEP       = $A2
 THEN       = $A0
 TO         = $9D
;
            * = $6800
;
;** 'LIST' PROMPT
            LDA #'l
            JSR OUTPUT
            LDA #'i
            JSR OUTPUT
            LDA #'s
            JSR OUTPUT
            LDA #'t
            JSR OUTPUT
;** INPUT LINE NOS.
            JSR INPUT
;** STORE LINE NOS.
            LDA #BUFFER
            STA BPTR
            LDA #0
            STA BPTR+1
            JSR CHRGOT
            BCC GETNUM
            BEQ GETNUM
            CMP #'-
            BNE EXIT
GETNUM      JSR EVAL
            JSR FINLIN
            JSR CHRGOT
            BEQ L1
            CMP #'-
            BNE EXIT
            JSR CHRGET
            JSR EVAL
            BNE EXIT
L1          LDA POKER
            ORA POKER+1
            BNE START
            LDA #$FF
            STA POKER
            STA POKER+1
START       LDY #1
            STY QUOFLG
            LDA (VARPTR),Y
            BEQ EXIT
            JSR CTRLC
            JSR LF:CR
            INY
            LDA (VARPTR),Y
            TAX
            INY
            LDA (VARPTR),Y
            CMP POKER+1
            BNE L2
            CPX POKER
            BEQ L3
L2          BCS EXIT
L3          STY YSAVE
            JSR NUMPRT
            LDY YSAVE
            LDA #SPACE
L4          JSR OUTPUT
            STA CHRSAV
            CMP #'"
            BNE L6
            LDA QUOFLG
            EOR #$FF
            STA QUOFLG
L6          INY
            LDA (VARPTR),Y
            BNE CHAR
            TAY
            LDA (VARPTR),Y
            TAX
            INY
            LDA (VARPTR),Y
            STX VARPTR
            STA VARPTR+1
            BNE START
EXIT        LDX #$FE
            TXS
            JMP OKAY
;
BRNCH1      BEQ L6
BRNCH2      BEQ L4
BRNCH3      BNE L4
BRANCH      BNE L6
;
CHAR        BPL COLON
            BIT QUOFLG
            BMI L4
TOKEN       STA TOKSAV
            CMP #TO
            BEQ S.TEST
            CMP #THEN
            BEQ S.TEST
            CMP #AND.
            BEQ S.TEST
            CMP #OR
            BEQ S.TEST
            CMP #STEP
            BNE NOSPC
S.TEST      LDA #SPACE
            CMP CHRSAV
            BEQ NOSPC
            STA CHRSAV
            JSR OUTPUT
NOSPC       LDA TOKSAV
            SEC
            SBC #$7F
            TAX
            STY YSAVE
            LDY #$FF
NEXTOK      DEX
            BEQ L5
NEXCHR      INY
            LDA TOKTBL,Y
            BPL NEXCHR
            BMI NEXTOK
L5          INY
            LDA TOKTBL,Y
            BMI ENDSPC
            JSR OUTPUT
            BNE L5
;
COLON       CMP #':
            BNE L4
            BIT QUOFLG
            BMI L4
            JSR OUTPUT
            INY
            LDA (VARPTR),Y
            CMP #SPACE
            BEQ BRNCH2
            DEY
ADDSPC      LDA #SPACE
            BNE BRNCH3
;
ENDSPC      AND #$7F
            JSR OUTPUT
            LDY YSAVE
            INY
            LDA (VARPTR),Y
            DEY
            CMP #SPACE
            BEQ BRNCH1
            LDA TOKSAV
            CMP #STEP+1
            BCC ADDSPC
            CMP #AND.
            BEQ ADDSPC
            CMP #OR
            BEQ ADDSPC
            BNE BRANCH
```

FOR SALE

POWER SUPPLY EX OSI C2P 110V AC input 4.5A @ 5V dc $30.00

110V 75mm cooling fan ex OSI C2P $17.00

240V/110V 100VA transformer in case $15.00
(Would be sent surface mail due to weight)

High Speed Cassette Interface. Will run at 1200 baud and connects to system via a PIA, VIA or similar. Connect one or two cassette players with motors under remote control if desired. Driver routine for 6800 CPU available which may assist 6502 users to interface. Documentation included. $90.00

For all these items contact Wayne Geary,

Superboard

Newsletter of the Ohio Superboard User Group, 146 York Street, Nundah, 4012.

SOFTWARE REVIEW - OSI Invaders.

This is OSI's version of the famous (some would say notorious) Space Invaders program which started the video games craze. The program is a mix of Basic and M/C programming. The Basic part gives game playing information and keeps the scores for each level of play, while the M/C part controls the game , screen and keyboard. The invaders themselves show very creative programming. By combining characters 242 through 247, a fascinating little creature with flapping "arms" is produced.

Rows of invaders march across the screen, bombing your gun and shields while you dodge and blaze away. Every time a row reaches one side of the screen, all the invaders drop down a level. The object of the game is to shoot them all before one lands. You gain one gun every time you clear a screenful. Level of play, the highest score at that level, and present score are continually updated on the screen above the invaders - not that you have much time to look during a game. As you thin the ranks out, the remaining invaders move faster and are harder to hit. The last invader is a regular little rocket!

There are fifteen levels of play - from impossibly fast to very slow. Practical limits are from 10 to about 3, depending on your reflexes.

To sum up - the best game of this type that I have seen. Excellent in every aspect. Highly recommended. Only available for the C1P as far as I know.

OSI Invaders is in the OSUG Library at the normal postage rates for Members.

GO INTO ORBIT

```
10 FOR A=1TO24:PRINT:NEXT:P2=6.28318:C=53743:B=32:P=232
20 R=9:I=.4
30 FOR A=I TO P2 STEP I:X=INT(R*COS(A)):Y=INT(R*SIN(A))*B
40 POKE C+X+Y,P:FOR T=0 TO 10:NEXT:POKE C+X+Y,B:NEXT:GOTO 30
```

For a 64 character screen, add L=B*2 at the end of line 10 and change the B in line 30 to L. Also change C to an address at the centre of your screen.

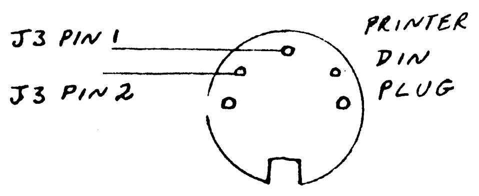
Variables:- P2 = 2 x PI, R= radius of orbit
I= interval between plots, T= time delay

QUICKPRINTER II INTERFACE by John Olding

Some time ago, Tandy offered the Quickprinter for sale at a very reasonable price. This machine has its limitations, printing on aluminised paper that is only three inches wide which makes it useless for wordprocessing applications but OK for program listings for error correction etc. The Quickprinter is easily connected to the Superboard. You need to have the RS-232 area populated but no negative swing is required. You also need 600 Baud. Switch to 1 MHz.

```
POKE 13,9:POKE 517,1:POKE 518,220:POKE 15,31
```

The connections are as shown at right:-
Viewed from end of plug to printer.



— SUPERBOARD —

COMPETITION

Just a short reminder that the programming competition closes on February 12th, 1983. Your entry form was included in the last KAOS Newsletter.

- -

BOOK REVIEW by Bernie Wills.

Microcomputer Design and Troubleshooting is 350 pages in soft cover, written by Eugene Zumchak and published by Howard Sams.

The book is hardware oriented, explaining the making of a minimum cost 6502 development system which could be used to produce dedicated controllers etc. The application of the Aim, Kim, and Sym systems to these tasks is also explored. A versatile Eprom programmer is described, complete with M/C software.
Various chapters cover timing, clocks, gate delays, bus control,buffering, address decoding, multiplexed keyboards, programmable ports, and serial and parallel input/output. There is an informative chapter on servicing microprocessor systems, static and dynamic board testing and emulation.

To summarise: If you need easily readable and very practical information on design, expansion, or modification of 6502 systems hardware, then this book will suit. Recommended price is $22.75

- -

COMPUTERS IN THE SCHOOL- A STUDENT'S VIEW by Ross Baker.

In September, 1980, Mitchelton High School in Queensland conducted a walk-a-thon to raise money for a computer system. The school purchased a PDP 11/03 system and a C1P. A special room was set aside as a computer room and air conditioned. The PDP 11/03 has as peripherals, a VDU, printer, card reader and dual disk drives. The card reader allows many students' programs to be checked quickly, and the programming language can be Cobol, Fortran, Pascal or Basic. The mini-computer is used mainly for school administration work, and prints out students' report cards at the end of each semester.

There are now six C1P's available for the use of students, and the computer room is open before school from 8 am to 8.45 am, during the lunch break, and after school until 4 pm. Students may book a computer twice each week, and any idle machines are available on a first come, first served basis.

Most of the machines are now fitted with switch selectable Baud and CPU rates, and with components added to the RS-232 interface, access to the PDP 11's printer and disk storage is possible. A single disk drive and expansion interface is expected to be fitted to our latest series II C1P soon.
For one period each week, grade 10 students have a computer awareness class. The course is structured to take some of the mystery out of computing. Subjects covered include the history of computing, job prospects in the computer field, programming in Basic, and use of the printer and C1P.

Grade 11 and 12 students have programming integrated into the maths 1 and 2 units. One example of an assignment was to write a Basic program to print the height and velocity of an object thrown into the air. A later addition to the assignment was to plot the positions at 1 second intervals using TAB statements.

Use of computers, and computer education in schools would appear to have a bright future.

- -

Next month, a review of BASIC 5, a chip with spectacular graphics functions.

Ed Richardson.

Queensland User Group Meeting.

Date: 12.12.82   Time: 5½ hours   Attendance: 24   Computers: 7

For the first time ever, disk systems outnumbered cassette based systems. Paul Brodie had his Shugart and Tasker Bus on display after much work on the system by himself and Robin Wells. A demo had been planned to show off the free modems, but it went awry, and more head scratching was in order.

Brendan Vowles had his Rabble and Tasan boards for inspection but couldn't give an active demo of the Rabble board functions. Most of the members were impressed by the finish of the Rabble board and it now seems to be the best way to go for expansion to disk.

David Hill offered printouts of any member's programs on his ICL printer, and spent most of the afternoon with it chattering happily.

Bill Simpson demonstrated some of the software developed for and by his school and himself with emphasis on education.

Robin Wells had some interesting M/C games simulating alien attacks on your spacecraft as seen from the cockpit. A lot of practice was needed to master the controls.

Alan Calvert showed us his Looky Video case which was very roomy and seemed excellent value for money. Every conceivable cut out has been made on the back panel to suit the keenest hardware modifier.

Bernie Wills had converted Sargon II Chess to use the chess characters programmed into his enhanced character set. Everyone agreed that at last we had a superb chess game and superb graphics - no mental character conversions being required to play the game on the screen. Bernie offered to supply the Sargon II + version to anyone who already has Sargon II Chess and had purchased one of his character sets. Just send him a tape plus 4 * 27¢ stamps to cover postage costs. I can guarantee that it improves the game dramaticly.

That was about it. I make no apologies for our unorganised form of meetings. I simply provide the time, place and refreshments, and the members make the meeting whatever they want it to be. Everyone seems to enjoy it.

Ed Richardson.

COMPUTER CLINIC

Carl Nielsen offers fast turnaround for repairs to microcomputers of all types including OSI at his Computer Clinic, 2 Forestlea St, Bracken Ridge, QLD 4017. Carl has twelve year's experience in the computer field.

Basic charge is $50 plus parts, however if extensive repairs are required the hourly rate is $20. Computer Clinic will also install modifications at a basic cost of $35. Phone Carl

EPSON PRINTER BUFFER

G.L. Wragg of offers a 32k or 64k memory board for the Epson MX-80 or MX-100 printers. The board plugs into the auxiliary interface connector inside the Epson without modification.

The advantage of such a system is that the microcomputer can dump the printer information into the buffer and then continue with other tasks while the board takes over the printing task. In the hobby field, the feature is not so important because time is usually not a problem. However for business use, a printer buffer would be a necessity. 32k costs $188 and 64k is $275. For more information, phone between 6 and 10pm.

MORE FROM THE WEST

by Wayne Geary

In Vol.3 No.2 there was a software review for Galaxia. I have modified a C1 version for my C2 and learnt a few things in the bargain. To change the speed alter location #04F0 which is normally set at #05. #01 to #09 give a good range of speeds. Note: #00 is DEAD slow.

Since my previous letter in Vol.3 No.2 I have written an Adventure game based on Star Wars. It runs in 16K and has 37 rooms/sections though some have the same name and are therefore not obvious.

I have just written a ROMable Auto Line program for BASIC programs which not only detects when to add a line number but will also retain the old line number if the last line was cancelled by a '@'. Auto-line hooks into the input routine, is available directly after a cold start and can be turned on and off. Line numbers can be reset to default values which are: first line = 100 and increment = 10, by ConTRoL-S. This routine requires 2 bytes in DABUG to be changed or Auto-line will not work when DABUG is enabled. Auto-line with initialisation requires 1/4K of memory. Anyone enquiring about either PRINT AT or AUTO-LINE will receive both as I am organising an assembler listing with both routines on it. The price is the same at $1.00 to cover costs.

I have put together, on paper, a number of changes I have made to the Extended Monitor. These include:-

1. A new start for EXMON which clears the screen (using the DABUG routine), sets the display to 64 characters wide, and put a heading onto the screen.
2. Further mods to the start of EXMON to provide some extra space to modify the Disassembly routine.
3. A modified Disassembly routine which gives 27 lines at a time and no spaces between each group of lines. My printer (an Anadex DP8000) cancells all input since the last 'Return' if no input is received for 10 seconds. The routine ensures no data is lost.
4. A space (9 in fact) between the hex bytes and the mnemonics to provide room to put in labels by hand later.
5. A new routine in place of the 8 byte hex dump ('Z' command) to provide a much needed block comparison with printout of any variations which are found.
6. A routine called by the 'U' command which automatically downloads the Assembler/Editor from ROM to RAM, clears the screen, puts a heading on the screen and jumps to the Assembler.
   Three bytes in areas other than those covered in the above must be changed to accomodate the changes to the start of EXMON and these are covered in the listing of the routine.

A listing of the changes I have made to the Extended Monitor is available from me, my address is 83 Second Ave, Rossmoyne, W.A. 6155, please enclose $1.00 to cover postage etc.

I am still making mods to my extended BASIC and as well as the PRINT AT and the AUTO LINE mentioned above I also have a DELete function which deletes a sequence of lines (particularly useful when you are using OSUG's BASIC4 and have a utility on top of your program). The DELete command has about one page of BASIC1 ROM altered and requires only 63 bytes external to the BASIC ROMs, it is in place of the LET command. I am also working on modifying a M/C line renumberer to run in ROM, using about 80 bytes less and callable from BASIC by ConTRoL-R.

THE BEGINNING MACHINE LANGUAGE PROGRAMMER....Part 7

by David Dodds

The most commmon technique for diverting the flow of a program is to use one of the conditional branches. Sometimes the diversion is outside the range of a branch instruction and other methods must be used. Multiple branching (branching to another branch) is one technique which can be used but is not usually very practical. This month I propose to explore 2 other methods for controlling the flow of a program.

The 6502 allows branching of any length (up to the addressing limits of the processor) with the JUMP instruction. The mnemonic for JUMP is JMP. JMP is most often used in the absolute addressing mode which is analogous in operation to GOTO in BASIC. In this mode the two bytes following the JMP opcode specify the value which is to be loaded into the program counter. If for example the instruction JMP $FE00 was processed then the program counter would be set to $FE00 resulting in a jump into the ROM monitor.

There is however a less commonly used form of the JMP instruction which uses INDIRECT addressing. JMP is the only instruction available in indirect mode. The two bytes following the JMP opcode still specify an absolute address but this is the address where the value for the program counter is stored.

In assembly language an indirect jump is written as

```
          JMP (address)
```

The process cannot be legally written in BASIC but would best be expressed as:

```
    GOTO(PEEK(A)+256*(PEEK(A+1))
```

The Superboard enters many of its routines such as SAVE, LOAD and CONTROL C using indirect jumps through the monitor ROM. SAVE for example is entered via the instruction JMP ($0220).
This feature makes it very easy to replace the standard system routines with user written routines. Unfortunately this facility is not available on standard C2 and C4 OSI computers.

Another common program control device is the subroutine call. Subroutines are an important part of a programming language and they are particularly important in assembly language.

In BASIC subroutines are called with GOSUB and terminated with RETURN. In assembly language the mnemonics JSR (Jump to SubRoutine) and RTS (ReTurn from Subroutine) perform the same function.

JSR is a three byte absolute mode instruction. Notice that the primary task of JSR is jump. However since the jump is to a subroutine the return address must be recorded for resetting the program counter when an RTS is executed.

The 6502 uses the stack to record return addresses. Since the program counter is pointing at the next instruction once the JSR data has been read, the CPU simply PUSHes this 2 byte value onto the stack before writing the subroutine address into the program counter. On encountering an RTS the CPU PULLs the top two bytes of data from the stack and loads them into the program counter.

One problem which can arise in using subroutines is that because there is only one stack, both the system and the programmer must share it. There is no way that the CPU can check whether the data it pulls is the correct address.

Many a program has crashed because data PUSHed onto the stack after a subroutine call has not been PULLed before an RTS is performed. Likewise crashes occur because return addresses have been PULLed by a subroutine causing the CPU to read incorrect data into the program counter.

Despite its dangers the subroutine remains an extremely powerful tool. Subroutines are most commonly used where the same operation needs to be performed at various places throughout a program. These programs then only need be written and tested once but may be used repeatedly.

When writing subroutines for machine language applications it is important to keep them simple. Avoid performing several tasks within a routine as this will lessen its usefulness. It is far better to write two or more simple routines which you can reuse, rather than one longer one which you will never use again.

There are many useful subroutines in the 65V monitor and BASIC-IN-ROM chips. Typically routines available include input, output, code conversion, interface and arithmetic processing. Many of the books available give details of what can be used. These routines work! Since there is no point in re-inventing wheels it makes sense to use them.

The following routines are especially handy and should be used freely to make your programming easier:

INPUT

$FFEB :-Calling this routine allows you to input a character from either tape or the keyboard depending on the (BASIC) load flag. The routine uses both index registers and the accumulator so if you need any of the values in them save them before making the call to $FFEB. The character is returned to you in the accumulator.
$FEED :-the keyboard part of $FFEB
$A357 :-This is the fill-the-input-buffer-until-carriage-return routine. Characters are obtained via $FFEB and stored from $13 onward. All system conventions such as auto repeat, backspace etc apply. The final carriage return is converted to a null (00) in the buffer.

OUTPUT

$FFEE :-This routine sends whatever is in the accumulator to the screen (and to tape if the save flag is set).
$BF2D :-is used by $FFEE to output the accumulator to the screen. Auto carriage return, line feed and screen scroll are generated automatically as required.
$A8C3 :-BASICS' message printer routine. Set the accumulator with the low byte of the absolute address of the start of your message, set Y with the high byte and call this routine to print it. The message must end with a null.

Modern developments in programming theory place a heavy emphasis on structured programming and top down design. As a result the subroutine has become more important. It is now commonplace to see the main line of a program consisting of very little more than a series of subroutine calls.

The following simple program demonstrates the point:

```
 10 ;TV TYPEWRITER PROGRAM
 20      * = $0222
 30 MESAGE = $A8C3
 40 READKY = $FEED
 50 ECHOIT = $BF2D
 60          LDY #$02
 70          LDA #$32
 80          JSR MESAGE
 90 LOOP     JSR READKY
100          JSR ECHOIT
110          JMP LOOP
120          .BYTE $0D
130          .BYTE 'TV TYPEWRITER'
140          .BYTE $0A,$0D
150          .BYTE 'READY',$0D
160          .BYTE $0A,$0A,0
```

In case you haven't got the general idea, lines 60 to 80 print a message on the screen, then the loop part of program waits for you to press a key and then writes the character to the video ram before jumping back to get the next character.

Over the next few months I plan to develop with you a series of subroutines which will eventually come together to form a graphics composer.

```
 10                  ;TV TYPEWRITER PROGRAM
 20 0222                  * = $0222
 30 A8C3=            MESAGE = $A8C3
 40 FEED=            READKY = $FEED
 50 BF2D=            ECHOIT = $BF2D
 60 0222 A002                 LDY #$02
 70 0224 A932                 LDA #$32
 80 0226 20C3A8               JSR MESAGE
 90 0229 20EDFE      LOOP     JSR READKY
100 022C 202DBF               JSR ECHOIT
110 022F 4C2902               JMP LOOP
120 0232 0D                   .BYTE $0D
130 0233 54                   .BYTE 'TV TYPEWRITER'
130 0234 56
130 0235 20
130 0236 54
130 0237 59
130 0238 50
130 0239 45
130 023A 57
130 023B 52
130 023C 49
130 023D 54
130 023E 45
130 023F 52
140 0240 0A                   .BYTE $0A,$0D
140 0241 0D
150 0242 52                   .BYTE 'READY',$0D
150 0243 45
150 0244 41
150 0245 44
150 0246 59
150 0247 0D
160 0248 0A                   .BYTE $0A,$0A,0
160 0249 0A
160 024A 00
```

DEAR PAUL

Q. I have installed a disk drive motor switch (Dodd's type) and find that frequently the drive motor switches itself on even though nothing has accessed the drive (eg. when the system is waiting an input from the keyboard). What could be the possible reasons for this? Note: I am using a Superboard with a Tasker Buss expansion system.
A. When I had a Tasker Buss system I noticed this effect, however now that I have a Rabble board this never happens. I assume then, that for some reason the Tasker disk controller board occasionally thinks that something has accessed $C000 which will turn the motor on, or my motor control switch occasionally triggers itself; why, I don't know. This effect never caused any problems, so I suggest that you either live with it or try building Bill Chilcott/Ray Gardiner's motor control circuit from the Rabble board manual.

Q. Is there a simple way to load BASIC programs from tape to disk using OS65D V3.2?
A. In theory, yes. Boot up, enter BASIC and type DISK!"IO 01" (no carriage return). Press play on your recorder and wait for the leader (the continuous tone) then press RETURN. The software should load until the 'OK' message and 'SN ERROR' will appear. The loading will stop (no need to press SPACE). Your problem is converting the cassette software to disk (all the pokes and so on will need to be changed)!

Q. Where does the standard line printer interface live on the OSI address map? I'm currently using the ACIA (device #1) but wish to claim this for use with a modem, and restore the printer to a PIA (device #4).

A. OSI's device #4 is a card which plugs into the 16 pin I/O bus. This card costs about $200, so I assume you don't want to use it! George and I at COMP-SOFT use a 'standard' COMP-SOFT device #4 driver with a PIA at location $C004. Port A is used for the printer data lines (PA0-PA7) PB0 is used for the strobe, and PB1 for the busy line.

Additions to the DOS are:

```
249F  8D04C0    STA $C004
24A2  CE06C0    DEC $C006
24A5  EE06C0    INC $C006
24A8  AD06C0    LDA $C006
24AB  2902      AND #$02
24AD  D0F9      BNE $24A8
24AF  60        RTS
```

And in your BEXEC*:

Continued page 14

```
xxx PI=49156:POKEPI,255:POKEPI+3,4:POKEPI+2,1:POKEPI+3,4:POKEPI+2,1
```

DISK FILES

by Graeme Reardon

So! You have finally tired of all those easy to get at disk games and have decided to get dinkum and write yourself a useful program to store and retrieve information. Well that should be super simple. After all you managed to do it on cassette and only had one page of instructions.

If you are like me, you then discovered the users manual is rather difficult to follow, and it certainly does not teach you (by itself) how to read and write files. So here is a simple program to show how to write some meaningful data to a file on disk and then retrieve it.

However, we must use the CREATE utility to create a file called (in this example) "XXXX". You can change the name of your file if you wish but change line 10 in the program.

Now run CREATE again and create a file to store the program listing that you are going to type in. You can also PUT the program onto the DATA file so that there is something for the progrram to look at when it is setting up the file initially but make sure that you answer (YES) in line 40 the first time you run the program.

Now run the program called "CHANGE". CHANGE will allocate space for your disk buffers so just answer the questions as prompted. Much of the above is unnecessary if you use COMP-DOS 1.2 - see notes in program. You could add more lines into the program at lines 130 to 151 as long as the total of each record did not exceed 128 in this case. For example "INFORMATION 1" could be changed to "NAME?" and "INFORMATION 2" changed to "ADDRESS?". A little bit of thought is required here and you will see that each word of information is added to the last (in line 150) before putting them to disk. The spaces between are to format for output.

There are many possibilities for expansion of this program including editing, sorting and updating - perhaps some of these could be discussed in future articles.

Next page please

```
1 REM DEMO FILE PROGRAMME by G. REARDON
10 DISK OPEN,6,"XXXX"  :REM  XXXX= FILE NAME
20 DISK GET,0
25 INPUT#6,HM$      :REM NUMBER OF RECORDS
27 HM=VAL(HM$)
28 PRINT"There are ";HM;"Entries in the file"
29 PRINT:PRINT:PRINT
30 INPUT"Do you wish to create a new file";Y$
40 IF ASC(Y$)<>89THEN90
50 DISK GET,0
60 HM$="0"
65 HM=0:  REM  NO RECORDS IN FILE
70 PRINT#6,HM$
80 DISK PUT
85 PRINT:PRINT:PRINT
90 PRINT"1    Input information"
92 PRINT"2    Print out information";
95 INPUTQ:IFQ=2THEN200
100 REM INPUT ENTRIES
110 HM=HM+1
115 PRINT:PRINT: PRINT"Type EXIT to finish entries"
120 DISK GET,HM
130 PRINT:PRINT:INPUT"INFORMATION 1";A1$:IF A1$="EXIT"THEN182
140 INPUT"INFORMATION 2";A2$
150 RE$=A1$+"    "+A2$: REM   COULD ADD MORE FIELDS - up to
151 REM  maximum of 128 characters per record
160 PRINT#6,RE$
170 DISK PUT
180 GOTO110
182 HM=HM-1 :HM$=STR$(HM)
183 DISK GET,0
185 PRINT#6,HM$:  REM  NEW NUMBER OF RECORDS
187 DISK PUT
200 REM PRINT OUT TO REQUIRED DEVICE
201 PRINT:PRINT:INPUT"Do you want printout to a printer";Y$
202 DV=2:IF ASC(Y$)=89THEN DV=1
205 REM  (DV=1) IS CASSETTE PORT OR PRINTER AT THAT PORT
210 IF HM=0THENPRINT:PRINT:PRINT#DV,"No information in file":GOTO300
220 FOR NO=1TOHM
230 DISK GET,NO
240 INPUT#6,RO$
250 PRINT:PRINT:PRINT#DV,RO$
260 PRINT#DV,
270 NEXT
280 PRINT:PRINT:PRINT#DV,"End of information in files"
300 DISK CLOSE,6
310 END
400 : To change record length add 15 POKE12042,48:POKE12076,6
410 : See earlier article in KAOS
420 : Add line 290 POKE12042,24:POKE12076,7
430 : Check these pokes for 8 inch systems
440 : If using COMP-DOS 1.2 add the following lines
450 : 5 DISK!"BS 06":CLEAR
460 : 305 DISK!"BR"
470 : I;f you have 1.2 you do not have to run 'CHANGE' but you
480 : still create a Data file using DISK!"CR XXXX,YY"
490 : XXXX= Any file name.   YY= Number of tracks
500 : then DISK!"ZE XXX"
```

NEW EX MON DISASSEMBLER (Cassette Version)
by Gerry walker

```
1FB7 201C1B             JSR $1B1C
1FBA A5DC               LDA $DC
1FBC 85D5               STA $D5
1FBE A5DD               LDA $DD
1FC0 85D6               STA $D6
1FC2 20071B             JSR $1B07
1FC5 A20A               LDX #$0A
1FC7 8E131A             STX $1A13
1FCA A204               LDX #$04
1FCC 20511C             JSR $1C51
1FCF CA                 DEX
1FD0 D0FA               BNE $1FCC
1FD2 20821A             JSR $1A82
1FD5 20B319             JSR $19B3
1FD8 20941A             JSR $1A94
1FDB 85D5               STA $D5
1FDD 84D6               STY $D6
1FDF 38                 SEC
1FE0 A5DE               LDA $DE
1FE2 E5D5               SBC $D5
1FE4 A5DF               LDA $DF
1FE6 E5D6               SBC $D6
1FE8 B0D8               BCS $1FC2
1FEA 20071B             JSR $1B07
1FED A904               LDA #$04
1FEF 8D131A             STA $1A13
1FF2 60                 RTS
```

The above program is a modification of Frank Halley's new disassembler (KAOS Vol.3 No.2) for use with the cassette version of the ExMon. I have found it to be such a vast improvement when printing disassemblies that I thought you may like to publish it as other readers might find it of use too.

It is located in the space occupied by the Z routine with ExMon at 1800H and operates in the same way as the original except that I have added the facility to specify separately the spacing before the start of code and between the code and mnemonics, the spacings are set by locations 1FC6 and 1FCB.

If you wish to retain the Z routine for use with the improved Search program recently published, as I have, then you will need to relocate it elsewhere, such as at 0FB7 or in EPROM.

Maybe some enterprising programmer will come up with a way of fitting them both into 2K.

FOR SALE

2 assembled and tested 8K static RAM cards for the Tasker Buss, without RAM, $20.00 ea. I will strap the cards for any block of memory requested in the upper or lower 32K boundaries.
1 Geoff Cohen floppy disk controller fully assembled but no chips on it, not tested, $10 or offer.
1 Geoff Cohen C1 to C4 video board assembled with sockets , not tested, no chips, $20 or offer.
Contact David Tasker

Dear Paul *Continued from page 10*

Q. I have a printer connected to my Tasker I/O board at address $E004. The printer works perfectly in the DOS, however I cannot get the DOS to boot unless the printer is switched on. Why?
A. I can think of two reasons, neither of which I can be sure of being the problem. Firstly $E000 - $E800 is the location for the colour RAM on OSI machines. Perhaps the DOS tries to clear these location, but finds that it can't. Secondly, OSI may have an IO device at this address and this causes your problem. Either way, I can not be sure, but I have never heard of this problem before.

Q. What does the 'WAIT' instruction do?
A. WAIT is generally used to poll (or check) an I/O port location. It has two forms:
i) WAIT I, J
ii) WAIT I, J, K

The first form is used the most. I is a memory location, J and K are integers between 0 and 255. The first form peeks memory location I, ANDs the result with J and keeps doing this until the result is not zero. For example, if you had an input port at location $C004, and had a push button connected to bit 0 (the first bit), and you wanted to pause until the button was pressed the instruction to use would be:

```
WAIT 49156,1
```

The second form peeks address I, exclusive ORs this with K and ANDs the result with J. This happens until the result is not zero. I have never used this form of the instruction.

Q. What are the differences between disk BASIC and ROM BASIC?
A. Disk BASIC (OS65D) does not have the reserved words LOAD and SAVE, and instead has the words DISK and EXIT. The EXIT command causes the computer to leave BASIC and enter the DOS (disk operating system). Disk is used to send commands to the DOS via the DISK! "doscmd" statement, and to handle file commands, DISK OPEN, DISK CLOSE, DISK GET, DISK PUT. The disk BASIC also has 9 digit real numbers instead of 6 digits and of course all the Peeks and Pokes are different. For more information see the BASIC REFERENCE MANUAL by OSI.

Here are some extracts from a letter sent to me by Mr Ian Davidson regarding data transmission and RS232.

"I would like to suggest some expansions to your answers in KAOS Vol.3 No.2 which should clarify some aspects of data transmission.

FULL DUPLEX: The ability to transmit in both directions simultaneously.
HALF DUPLEX: The ability to transmit in both directions but not at the same time."

He then goes on to explain 'echo-plex' and to explain why some systems use Full Duplex and why others use Half Duplex. He also says something about RS232:

"...RS232 is not a data transmission standard, but rather a serial data interface standard for connecting to data communication equipment (modems). The transmission standards are defined elsewhere."

Thank you Ian Davidson.

Hardware for the Tasker Buss

A 24K memory board, standard size , that can take 12 assorted types of 2K chips. This board has been tested at 1MHz for 6116 CMOS RAM, 2128 static RAM and 2716 EPROMs. The board may have a combination of any of these chips. Power consumption approx. 2W. The board is double sided but as I do not anticipate a large demand for the board as many people will have established memory or a Rabble board, I will not be getting plated through boards. Because the board requires soldering on both sides and sockets are difficult to solder on top of the board, I will not charge for assembly at this time. I will assemble and test each board with all components and provide sockets for the memory. The board is $25, parts and sockets $17, postage $2. There will be an assembly charge of $20 after the 1st March. Limit of 2 boards per customer. Notes and circuits with strapping applications are available, please send $1 to cover postage.
Thanks to David Anear the 16 pin I/O buss for the Tasker Buss (refer circuit in KAOS Nov. 1982) is finished. At the moment the only board we have that will plug into it is the new EPROM programmer. Notes and application hints are being prepared and will be available soon.

Contact David Tasker

R65C02, R65C102, R65C112 Microprocessors

INSTRUCTION SET ALPHABETIC SEQUENCE

NOTES:
(1) New Instruction
(2) Previous Instruction with additional addressing mode(s)

Mnemonic	Function	Mnemonic	Function
(2) ADC	Add Memory to Accumulator with Carry	NOP	No Operation
(2) AND	"AND" Memory with Accumulator		
ASL	Shift Left One Bit (Memory or Accumulator)	(2) ORA	"OR" Memory with Accumlator
(1) BBR	Branch on Bit Reset	PHA	Push Accumulator on Stack
(1) BBS	Branch on Bit Set	PHP	Push Processor Status on Stack
BCC	Branch on Carry Clear	(1) PHX	Push X Register on Stack
BCS	Branch on Carry Set	(1) PHY	Push Y Register on Stack
BEQ	Branch on Result Zero	PLA	Pull Accumulator from Stack
(2) BIT	Test Bits in Memory with Accumulator	PLP	Pull Processor Status from Stack
BMI	Branch on Result Minus	(1) PLX	Pull X Register from Stack
BNE	Branch on Result not Zero	(1) PLY	Pull Y Register from Stack
BPL	Branch on Result Plus		
(1) BRA	Branch Always	(1) RMB	Reset Memory Bit
BRK	Force Break	ROL	Rotate One Bit Left (Memory or Accumulator)
BVC	Branch on Overflow Clear	ROR	Rotate One Bit Right (Memory or Accumulator)
BVS	Branch on Overflow Set	RTI	Return from Interrupt
		RTS	Return from Subroutine
CLC	Clear Carry Flag		
CLD	Clear Decimal Mode	SBC	Subtract Memory from Accumulator with Borrow
CLI	Clear Interrupt Disable Bit	SEC	Set Carry Flag
CLV	Clear Overflow Flag	SED	Set Decimal Mode
(2) CMP	Compare Memory and Accumulator	SEI	Set Interrupt Disable Status
CPX	Compare Memory and Index X	(1) SMB	Set Memory Bit
CPY	Compare Memory and Index Y	(2) STA	Store Accumulator in Memory
		STX	Store Index X in Memory
(2) DEC	Decrement Memory by One	STY	Store Index Y in Memory
DEX	Decrement Index X by One	(1) STZ	Store Zero
DEY	Decrement Index Y by One		
		TAX	Transfer Accumulator to Index X
(2) EOR	"Exclusive-OR" Memory with Accumulator	TAY	Transfer Accumulator to Index Y
		(1) TRB	Test and Reset Bits
(2) INC	Increment Memory by One	(1) TSB	Test and Set Bits
INX	Increment Index X by One	TSX	Transfer Stack Pointer to Index X
INY	Increment Index Y by One	TXA	Transfer Index X to Accumulator
		TXS	Transfer Index X to Stack Register
(2) JMP	Jump to New Location	TYA	Transfer Index Y to Accumulator
JSR	Jump to New Location Saving Return Address		
(2) LDA	Load Accumulator with Memory		
LDX	Load Index X with Memory		
LDY	Load Index Y with Memory		
LSR	Shift One Bit Right (Memory or Accumulator)		

R65C00 MICROPROCESSORS

FEATURES

- CMOS silicon gate technology
- Low Power (4MA/MHz)
- Downward software compatible with R6502
  - —Twelve additional instructions
  - —Two new addressing modes
- Single 5V ±20% power supply
- Eight bit parallel processing
- Decimal and binary arithmetic
- True indexing capability
- Programmable stack pointer
- Interrupt capability
- Non-maskable interrupt
- Use with any type of speed memory
- Eight-bit Bidirectional Data Bus
- Addressable memory range of up to 64K bytes
- "Ready" input
- Direct Memory Access capability
- Memory Lock Output
- 2MHz, 3MHz, and 4MHz versions
- Choice of external or on-chip clocks
- On-the-chip clock options
  - —External single clock input
  - —Direct Crystal Input (÷ 4)

MSD \ LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BRK Implied 1 7	ORA (IND, X) 2 6			□ TSB ZP 2 5	ORA ZP 2 3	ASL ZP 2 5	□ RMB0 ZP 2 5	PHP Implied 1 3	ORA IMM 2 2	ASL Accum 1 2		□ TSB ABS 3 6	ORA ABS 3 4	ASL ABS 3 6	□ BBR0 ZP 3 5**
1	BPL Relative 2 2**	ORA (IND), Y 2 5*	□ ORA (IND) 2 5		□ TRB ZP 2 5	ORA ZP, X 2 4	ASL ZP, X 2 6	□ RMB1 ZP 2 5	CLC Implied 1 2	ORA ABS, Y 3 4*	□ INC Accum 1 2		□ TRB ABS 3 6	ORA ABS, X 3 4*	ASL ABS, X 3 7	□ BBR1 ZP 3 5**
2	JSR Absolute 3 6	AND (IND, X) 2 6			BIT ZP 2 3	AND ZP 2 3	ROL ZP 2 5	□ RMB2 ZP 2 5	PLP Implied 1 4	AND IMM 2 2	ROL Accum 1 2		BIT ABS 3 4	AND ABS 3 4	ROL ABS 3 6	□ BBR2 ZP 3 5**
3	BMI Relative 2 2**	AND (IND), Y 2 5*	□ AND (IND) 2 5		□ BIT ZP, X 2 4	AND ZP, X 2 4	ROL ZP, X 2 6	□ RMB3 ZP 2 5	SEC Implied 1 2	AND ABS, Y 3 4*	□ DEC Accum 1 2		□ BIT ABS, X 3 4*	AND ABS, X 3 4*	ROL ABS, X 3 7	□ BBR3 ZP 3 5**
4	RTI Implied 1 6	EOR (IND, X) 2 6				EOR ZP 2 3	LSR ZP 2 5	□ RMB4 ZP 2 5	PHA Implied 1 3	EOR IMM 2 2	LSR Accum 1 2		JMP ABS 3 3	EOR ABS 3 4	LSR ABS 3 6	□ BBR4 ZP 3 5**
5	BVC Relative 2 2**	EOR (IND), Y 2 5*	□ EOR (IND) 2 5			EOR ZP, X 2 4	LSR ZP, X 2 6	□ RMB5 ZP 2 5	CLI Implied 1 2	EOR ABS, Y 3 4*	□ PHY Implied 1 2			EOR ABS, X 3 4*	LSR ABS, X 3 7	□ BBR5 ZP 3 5**
6	RTS Implied 1 6	ADC (IND, X) 2 6†			□ STZ ZP 2 3	ADC ZP 2 3†	ROR ZP 2 5	□ RMB6 ZP 2 5	PLA Implied 1 4	ADC IMM 2 2†	ROR Accum 1 2		JMP Indirect 3 5	ADC ABS 3 4†	ROR ABS 3 6	□ BBR6 ZP 3 5**
7	BVS Relative 2 2**	ADC (IND), Y 2 5*†	□ ADC (IND) 2 5†		□ STZ ZP, X 2 4	ADC ZP, X 2 4†	ROR ZP, X 2 6	□ RMB7 ZP 2 5	SEI Implied 1 2	ADC ABS, Y 3 4*†	□ PLY Implied 1 2		□ JMP (IND), X 3 6	ADC ABS, X 3 4*†	ROR ABS, X 3 7	□ BBR7 ZP 3 5**
8	□ BRA Relative 2 3	STA (IND, X) 2 6			STY ZP 2 3	STA ZP 2 3	STX ZP 2 3	□ SMB0 ZP 2 5	DEY Implied 1 2	□ BIT IMM 2 2	TXA Implied 1 2		STY ABS 3 4	STA ABS 3 4	STX ABS 3 4	□ BBS0 ZP 3 5**
9	BCC Relative 2 2**	STA (IND), Y 2 6	□ STA (IND) 2 6		STY ZP, X 2 4	STA ZP, X 2 4	STX ZP, Y 2 4	□ SMB1 ZP 2 5	TYA Implied 1 2	STA ABS, Y 3 5	TXS Implied 1 2		□ STZ ABS 3 4	STA ABS, X 3 5	□ STZ ABS, X 3 5	□ BBS1 ZP 3 5**
A	LDY IMM 2 2	LDA (IND, X) 2 6	LDX IMM 2 2		LDY ZP 2 3	LDA ZP 2 3	LDX ZP 2 3	□ SMB2 ZP 2 5	TAY Implied 1 2	LDA IMM 2 2	TAX Implied 1 2		LDY ABS 3 4	LDA ABS 3 4	LDX ABS 3 4	□ BBS2 ZP 3 5**
B	BCS Relative 2 2**	LDA (IND), Y 2 5*	□ LDA (IND) 2 5		LDY ZP, X 2 4	LDA ZP, X 2 4	LDX ZP, Y 2 4	□ SMB3 ZP 2 5	CLV Implied 1 2	LDA ABS, Y 3 4*	TSX Implied 1 2		LDY ABS, X 3 4*	LDA ABS, X 3 4*	LDX ABS, Y 3 4*	□ BBS3 ZP 3 5**
C	CPY IMM 2 2	CMP (IND, X) 2 6			CPY ZP 2 3	CMP ZP 2 3	DEC ZP 2 5	□ SMB4 ZP 2 5	INY Implied 1 2	CMP IMM 2 2	DEX Implied 1 2		CPY ABS 3 4	CMP ABS 3 4	DEC ABS 3 6	□ BBS4 ZP 3 5**
D	BNE Relative 2 2**	CMP (IND), Y 2 5*	□ CMP (IND) 2 5			CMP ZP, X 2 4	DEC ZP, X 2 6	□ SMB5 ZP 2 5	CLD Implied 1 2	CMP ABS, Y 3 4*	□ PHX Implied 1 2			CMP ABS, X 3 4*	DEC ABS, X 3 7	□ BBS5 ZP 3 5**
E	CPX IMM 2 2	SBC (IND, X) 2 6†			CPX ZP 2 3	SBC ZP 2 3†	INC ZP 2 5	□ SMB6 ZP 2 5	INX Implied 1 2	SBC IMM 2 2†	NOP Implied 1 2		CPX ABS 3 4	SBC ABS 3 4†	INC ABS 3 6	□ BBS6 ZP 3 5**
F	BEQ Relative 2 2**	SBC (IND), Y 2 5*†	□ SBC (IND) 2 5†			SBC ZP, X 2 4†	INC ZP, X 2 6	□ SMB7 ZP 2 5	SED Implied 1 2	SBC ABS, Y 3 4*†	□ PLX Implied 1 2			SBC ABS, X 3 4*†	INC ABS, X 3 7	□ BBS7 ZP 3 5**

□ — New Opcode

	0	
0	BRK	—OP Code
	Implied	—Addressing Mode
	1 7	—Instruction Bytes; Machine Cycles

†Add 1 to N if in decimal mode.
*Add 1 to N if page boundary is crossed.
**Add 1 to N if branch occurs to same page;
Add 2 to N if branch occurs to different page.